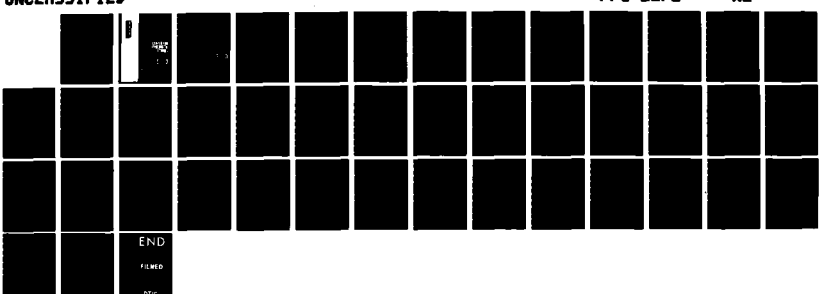
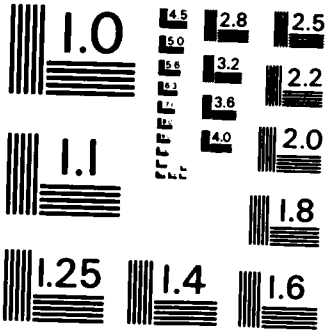
AD-A159 594 THE EQUAL FLOW PROBLEM(U) TEXAS UNIV AT AUSTIN CENTER FOR CYBERNETIC STUDIES I ALI ET AL. APR 85 CCS-RR-511 N00014-82-K-0295 1/1

UNCLASSIFIED F/G 12/2 NL





(2)

Research Report 511

THE EQUAL FLOW PROBLEM

by

Iqbal Ali
Jeffrey Kennington*
Bala Shetty*

CENTER FOR CYBERNETIC STUDIES
The University of Texas
Austin, Texas 78712

Research Report 511

THE EQUAL FLOW PROBLEM

by

Iqbal Ali
Jeffrey Kennington*
Bala Shetty*

April 1985

*Southern Methodist University, Department of Operations Research

This research was partly supported by ONR Contract N0014-82-K-0295 and by the Air Force Office of Scientific Research under Contract Number AFOSR 83-0278 with Southern Methodist University. Reproduction in whole or in part is permitted for any purpose of the United States Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
College of Business Administration 5.202
The University of Texas at Austin
Austin, TX 78712-1177
(512) 471-1821

ABSTRACT

This paper presents a new algorithm to solve a network problem with equal flow side constraints. The proposed solution technique is motivated by the desire to exploit the special structure of the side constraints and to maintain as much of the characteristics of pure network problems as possible. Not only has specialized software for the efficient solution of pure networks been developed, but the same computational efficacies lend themselves to the solution of sequences of minimum cost network flow problems by using reoptimization procedures. Our solution technique for the equal flow problem consists of solving two sequences of pure network problems. One sequence yields tighter lower bounds on the optimal value by considering the Lagrangean relaxation of the equal flow problem in which the side constraints are dualized. The second sequence yields upper bounds on the optimal value for the problem and maintains a feasible solution at all times. This sequence is obtained by considering a reformulation of the equal flow problem based on parametric changes in the requirements vector. The procedure has the added attractive feature that it provides a feasible solution which is known to be within a percentage of the optimal at all times. As such, the algorithm terminates when a solution with a prespecified tolerance on the objective function value is obtained. On NETGEN problems, using the first 150 arcs to form 75 equal flow side constraints, we found that the new algorithm is approximately 3 times faster than existing techniques and requires only 50% of the storage.

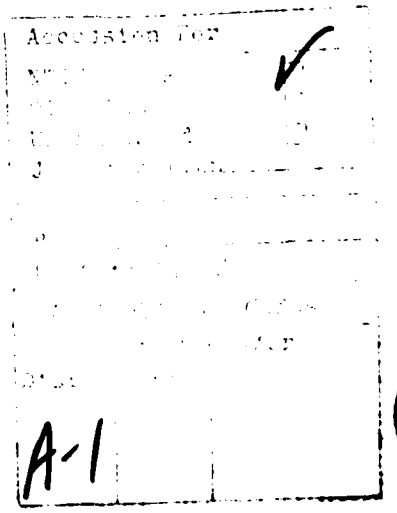
KEY WORDS

Linear Programming

Network Models

Networks With Side Constraints

Equal Flow Problem



I. INTRODUCTION

This paper presents a new technique to solve the equal flow problem. This problem is easily conceptualized as a minimal cost network flow problem with additional constraints on certain pairs of arcs. Specifically, given pairs of arcs are required to take on the same value. Applications of this model include crew scheduling [6], estimating driver costs for transit operations [28], and the two duty period scheduling problem [25]. The equal flow problem may be solved using a specialization of the simplex method for networks with side constraints. However, by exploiting the special structure of the side constraints, we have developed a new algorithm which results in a decrease in both computer storage and computation time.

It is well documented that pure network problems can be solved from fifty to one hundred times faster using specialized primal simplex software as compared to general linear programming systems. Motivated by this great advantage, our procedure solves the equal flow problem as a sequence of pure network problems and totally eliminates the need to deal with a basis matrix.

1.1 Problem Description

The equal flow problem is defined on a network represented by an (m,n) node-arc incidence matrix, A, in which K pairs of arcs are identified and required to have equal flow. Mathematically, this is expressed as:

$$\text{Minimize} \quad cx$$

$$\text{s.t.} \quad Ax = b$$

$$x_k = x_{k+K}, \qquad k = 1, \ldots, K$$

$$0 \leq x \leq u$$

where, c is a 1 x n vector of unit costs, b is an m x 1 vector of node requirements, 0 is an n x 1 vector of zeroes, x is an n x 1 vector of decision variables, and u is an n x 1 vector of upper bounds. The above definition, henceforth referred to as P1, assumes that the first 2K arcs appear in the equal flow constraints. This assumption is in no way restrictive since, by rearranging the order of the arcs, any equal flow problem with K pairs can be expressed in the above form. Note that the K pairs of arcs are mutually exclusive, i.e. an arc appears in at most one side constraint.

1.2 Survey of Related Literature

In 1961, Charnes and Cooper [7] presented a specialized algorithm for the model:

$$\text{Minimize} \quad cx$$

$$\text{s.t.} \quad Ax = b$$

$$Cx = d$$

$$x \geq 0,$$

where A and C are some general matrices but A has some favored structure. Their algorithm, called the double reverse method, takes advantage of the special structure of the matrix A. Variations of this algorithm may be found in [2, 8, 10, 15, 19, 24]. Specializations for multicommodity

problems may be found in [14, 20, 21].

In 1980, Shepardson and Marsten [25] showed that the two duty period scheduling problem can be reformulated as a single duty period scheduling problem with equal flow side constraints. They obtain a Lagrangean dual for this equal flow problem, by dualizing with respect to the equal flow side constraints. This Lagrangean dual is maximized using the subgradient optimization technique. In 1984, Turnquist and Malandraki [28] modeled the problem of estimating driver costs for transit operations as an integer equal flow problem. They obtain a Lagrangean dual for their problem, by dualizing with respect to the side constraints. Their algorithm is a slight modification of the subgradient optimization technique. They perform a line search between two successive solutions obtained during the subgradient optimization process.

Beck, Lasdon, and Engquist [5] transformed the equal flow problem into a quadratic programming problem which has a penalty for violating the equal flow constraints. They solved this nonlinear programming problem using the Fletcher-Reeves conjugate gradient method [9], a successive linear programming code [13], and a convex simplex code. If the penalty is sufficiently large, this approach is guaranteed to converge to the optimal solution of the equal flow problem.

1.3 Objective of the Investigation

The objective of this investigation is to develop and computationally test a new algorithm for the equal flow problem. This algorithm utilizes the subgradient optimization technique and is based on the relaxation/

restriction procedure proposed by Glover, Glover, and Martinson [11] for a generalized network model with special side constraints. We establish that the equal flow problem may be solved as two sequences of pure network problems, one sequence corresponds to computing a lower bound while the other corresponds to computing an upper bound. In the limit, both bounds will converge to the optimal objective value. Our implementation terminates when the difference between the bounds is within a prespecified tolerance.

The subgradient optimization technique requires the computation of subgradients, choice of appropriate step sizes, and the application of a projection operation. We show that the subgradients for the upper bound can be computed using the optimal dual variables obtained by solving pure network problems. We also develop theoretical results that yield an easy implementation of the projection operation. The step sizes selected are a modification of the ones proposed by Polyak [23]. For this choice of step sizes, we prove that our algorithm must necessarily obtain an iterate at which the objective value is arbitrarily close to the optimal objective value. In a computational study, comparing our code with a code that is designed to solve network problems with side constraints, we found that the new code runs approximately 3 times faster and requires 50% less core storage.

II. THE SUBGRADIENT ALGORITHM

The Subgradient Algorithm was first introduced by Shor [27] and is a general procedure for solving nonlinear programming problems. It may be viewed as a generalization of the steepest descent (ascent) method for convex (concave) problems in which the gradient may not exist everywhere. The subgradient is simply substituted in place of the gradient for those points for which the gradient does not exist. When this occurs, the algorithm may move to a point with objective value worse than the current point. Hence, the objective function does not necessarily improve at each iteration and consequently the convergence results of Zangwill [29] do not apply. Remarkably though, under fairly minor conditions on the step size, convergence can be guaranteed.

Let the nonlinear program PO be given by:

Minimize $f(y)$

s.t. $y \in G$

where f is a real valued function that is convex over the compact, convex, and nonempty set G. A vector η will be called a <u>subgradient</u> of f at $\bar{y}$ if $f(y) - f(\bar{y}) \geq \eta(y - \bar{y})$ for all $y \in G$. For any $\bar{y} \in G$, we denote the set of all subgradients of f at $\bar{y}$ by $\partial f(\bar{y})$. The subgradient algorithm makes use of an operation called the projection operation. The projection of a point x onto G, denoted by $P[x]$, is defined to be the unique point $y \in G$ that is nearest to x with respect to the Euclidean norm. Using the projection operation, we now present the subgradient algorithm in its most general form.

ALG 1 SUBGRADIENT OPTIMIZATION ALGORITHM

Step 0 (Initialization)

Let y_0 be any element of G, select a set of step sizes $s_0, s_1, s_2, \ldots$, and set $i \leftarrow 0$.

Step 1 (Find Subgradient)

Let $\eta_i \in \partial f(y_i)$. If $\eta_i = 0$, then terminate with y_i optimal.

Step 2 (Move to New Point)

Set $y_{i+1} \leftarrow P[y_i - s_i \eta_i]$, set $i \leftarrow i + 1$ and return to step 1.

Various proposals have been offered for the selection of the step sizes. Three general schema which have been suggested are:

i) $s_i = \lambda_i$,

ii) $s_i = \dfrac{\lambda_i}{\|\eta_i\|^2}$,

iii) $s_i = \dfrac{\lambda_i (f(y_i) - f^*)}{\|\eta_i\|^2}$, $0 < \lambda_i < 2$,

where f^* is the optimal value of f over G. If the constants, λ_i's, satisfy the following conditions:

$\lambda_i \geq 0$, all i; $\lim_{i \to \infty} \lambda_i = 0$; and $\sum \lambda_i = \infty$,

then the convergence of the algorithm is guaranteed using (i) or (ii) (see Goffin [12], Helgason [17], Kennington and Helgason [21]). For the

upper bounds, we use a modification of the third step size. The following result is available for this scheme.

Proposition 1 (Polyak [23])

Let f be a real valued convex function over the compact, convex, and nonempty set G. Also, let f* be the minimum of f and $\|\eta_i\| \leq C$ for all i and some constant C. Then there exists a $y_i^* \in G$ with $f(y_i^*) \to f^*$, if scheme (iii) is used.

Note that in (iii), f* is the optimal value of f over G. Since the optimal objective is unknown before solving the problem, we use a lower bound on f* in our implementation.

III. THE LOWER BOUND

Recall that the equal flow problem, which we denote by P1, is given by:

$$\text{Minimize} \quad cx$$

$$\text{s.t.} \quad Ax = b$$

$$x_k = x_{K+k}, \quad k = 1,\ldots,K$$

$$0 \leq x \leq u.$$

In our algorithm for P1, lower bounds on the optimal objective of P1 are used for step sizes and for termination. In this section, we describe a procedure to obtain these lower bounds.

Consider the following Lagrangean dual for P1, which we shall refer to as D1:

Maximize $h(w)$, where $w = [w_1,\ldots,w_K] \in R^K$, and

$$h(w) = \text{Min}\{cx + \sum_{k=1}^{K} w_k(x_k - x_{K+k}) : Ax = b,\ 0 \leq x \leq u\}.$$

Proposition 2 (Shetty [26])

Let $\bar{x}$ be a feasible solution to P1 and let $w = [w_1,\ldots,w_k]$ be a feasible solution to D1. Then $c\bar{x} \geq h(w)$.

Proposition 3 (Bazaraa and Shetty [4])

If P1 has a minimum, then the optimal objectives for P1 and D1 are equal.

As a consequence of Propositions 2 and 3, we may solve D1 to obtain a lower bound. We will now show that D1 may be solved using the

VI. COMPUTATIONAL EXPERIMENTATION

This section describes the computer implementation, EQFLO, and testing of our algorithm for the equal flow problem. The algorithm was tested on a set of 35 test problems randomly generated using NETGEN [22]. Computation times are compared with those of NETSIDE [2], a general purpose code for network problems with side constraints. Both NETSIDE and EQFLO are written in standard FORTRAN for an incore implementation and have not been tailored to either the machine or FORTRAN compiler used for testing.

6.1 Description of the Computer Codes

NETSIDE was developed by Barr, Farhangian, and Kennington at Southern Methodist University, Dallas, Texas. Designed to solve network problems with side constraints, it used a specialization of the revised simplex method known as the primal partitioning algorithm [15]. The basis inverse is maintained as a rooted spanning tree and a working basis inverse in product form. The reinversion routine is a modification of the work of Hellerman and Rarick [18] and uses the "spike swapping theory" of Helgason and Kennington [16]. The initial working basis consists of a combination of artificial and slack variables. The working basis is reinverted every 50 iterations. The pricing routine uses a candidate list of size 10 with a block size of 400. Both pricing and pivot tolerance are 1.E-6.

EQFLO is our implementaiton of ALG 4, and makes use of MODFLO [1] to solve pure network subproblems. MODFLO is a set of subroutines which may

Step 2 (Compute Upper Bounds)

2a. Set T ← 0, set IFLAG ← 1.

2b. Call Alg 3 (steps 2 and 3a)

2c. Set R ← R+1.

If R < ITERU, then go to 2b.

Step 3

Set R ← 0.

Set p ← p_0.

Go to 1.

In the above algorithm, IFLAG is used in obtaining a starting y from the solutions in 1a. The bases used in steps 1 and 2 are generated from the optimal bases obtained in the previous iterations.

V. THE ALGORITHM

In this section, we present our new algorithm for solving the equal flow problem. Let ITERL denote the number of iterations spent in step 1a in computing the lower bound before returning to the upper bound, and ITERU denote the number of iterations spent in computing the upper bound before returning to the lower bound. Also, let T denote the iteration count for the lower bound, R denote the iteration count for upper bound, and p_0 denote the initial step size for the lower bound.

ALG 4 SUBGRADIENT OPTIMIZATION ALGORITHM

FOR THE EQUAL FLOW PROBLEM

Step 0 (Initialization)

Initialize ITERL, ITERU, REREQ, λ_0, p_0, and tolerance ε.

Set $T \leftarrow 0$, set $Q \leftarrow 0$, set $R \leftarrow 0$, set $w \leftarrow 0$, and set IFLAG $\leftarrow 0$.

Set UBND $\leftarrow +\infty$, set LBND $\leftarrow -\infty$, set $p \leftarrow p_0$, and set $\bar{\lambda}_k \leftarrow \lambda_0$ for $k=1,\ldots,K$.

Set $\tilde{u} \leftarrow (\min(u_1, u_{K+1}),\ldots,\min(u_K, u_{2K}))$.

Step 1 (Compute Lower Bounds)

1a. Call ALG 2 (steps 2 and 3a).

1b. Set $T \leftarrow T+1$

If $T \leftarrow$ ITERL, then go to 1.

1c. (Initialize y)

If IFLAG $\neq 0$, then go to 2; otherwise,

set $y \leftarrow [\min(\tilde{u}_1, (\bar{x}_1 + \bar{x}_{K+1})/2),\ldots,\min(\tilde{u}_K, (\bar{x}_K + \bar{x}_{2K})/2)]$.

$$= \frac{\delta}{(2\lambda_i - \lambda_i^2)} + g^*\left(\frac{2}{2-\beta} - \frac{\beta}{2-\beta}\alpha\right)$$

$$\leq M\delta + g^*\left(\frac{2}{2-\beta} - \frac{\beta}{2-\beta}\alpha\right), \text{ where } M \text{ is a constant less than } \frac{1}{(2\varepsilon-\varepsilon^2)}.$$

This completes the proof of Proposition 12.

We can choose an integer N large enough that

$$\frac{c^2 \| y_1 - y^* \|^2}{(g^* - \bar{g})\delta} < N.$$

Adding together the inequalities obtained from (1) by letting i take on all values from 1 to N, we obtain

$$\| y_N - y^* \|^2 \leq \| y_i - y^* \|^2 - \frac{N(g^* - \bar{g})\delta}{c^2} < 0,$$

a contradiction. This justifies our assertion.

By simplifying our assertion further we get,

$$g(y_i) \leq \frac{\delta}{(2\lambda_i - \lambda_i^2)} - \frac{\bar{g}\,\lambda_i}{(2 - \lambda_i)} + \frac{2g^*}{(2 - \lambda_i)}$$

$$\leq \frac{\delta}{(2\lambda_i - \lambda_i^2)} - \frac{\alpha\, g^* \lambda_i}{(2 - \lambda_i)} + \frac{2g^*}{(2 - \lambda_i)}$$

$$= \frac{\delta}{(2\lambda_i - \lambda_1^2)} + g^* \left[\frac{2 - \alpha\,\lambda_i}{2 - \lambda_i}\right]$$

$$= \frac{\delta}{(2\lambda_i - \lambda_i^2)} + g^*\left(1 + \frac{\lambda_i}{(2 - \lambda_i)}(1 - \alpha)\right)$$

$$\leq \frac{\delta}{(2\lambda_i - \lambda_i^2)} + g^*\left(1 + \frac{\beta}{2-\beta}(1 - \alpha)\right)$$

Suppose that for all i,

$$g(y_i) > \frac{\lambda_i^2 \bar{g}}{(\lambda_i^2 - 2\lambda_i)} - \frac{2\lambda_i g^*}{(\lambda_i^2 - 2\lambda_i)} - \frac{\delta}{(\lambda_i^2 - 2\lambda_i)}, \text{ where}$$

$\delta > 0$ is given. Let $y^* \in S$ be an optimal point. By Proposition 11,

$$\| y_{i+1} - y^* \|^2 \leq \| y_i - y^* \|^2 + \frac{\lambda_i^2 (g(y_i) - \bar{g})^2}{\| \eta_i \|^2} + \frac{2\lambda_i (g(y_i) - \bar{g})\, \eta_i (y^* - y_i)}{\| \eta_i \|^2}$$

Since $\eta_i \in \partial g(y_i)$, $\eta_i (y^* - y_i) \leq g^* - g(y_i)$.

Thus,

$$\| y_{i+1} - y^* \|^2 \leq \| y_i - y^* \|^2 + \frac{\lambda_i^2 (g(y_i) - \bar{g})^2}{\| \eta_i \|^2} + \frac{2\lambda_i (g(y_i) - \bar{g})(g^* - g(y_i))}{\| \eta_i \|^2}$$

$$= \| y_i - y^* \|^2 + (g(y_i) - \bar{g}) \left[\frac{g(y_i)(\lambda_i^2 - 2\lambda_i) - \lambda_i^2 \bar{g} + 2\lambda_i g^*}{\| \eta_i \|^2} \right]$$

$$\leq \| y_i - y^* \|^2 - \frac{(g(y_i) - \bar{g})\delta}{\| \eta_i \|^2}$$

$$\leq \| y_i - y^* \|^2 - \frac{(g^* - \bar{g})\delta}{c^2} \qquad (1)$$

In our implementation, we use $\bar{g}$, a lower bound on g, in place of g*. The following propositions demonstrate that for $\bar{g}$'s close to g*, our procedure must necessarily obtain an iterate at which g is arbitrarily close to g*.

Proposition 11 (Kennington and Helgason [21])

If $\eta_i \neq 0$,

$$\| y_{i+1} - y_i \|^2 \leq \| y_i - y \|^2 + s_i^2 \| \eta_i \|^2 + 2 s_i \eta_i (y - y_i) \text{ for any } y \in S$$

and step size s_i.

Proposition 12

Let g* be the optimal value of g, and also let

i) $\alpha g^* \leq \bar{g} \leq g^*$, $0 \leq \alpha \leq 1$,

ii) $s_i = \lambda_i (g(y_i) - \bar{g}) / \| \eta_i \|^2$, and

iii) $0 < \varepsilon \leq \lambda_i \leq \beta < 2$ for all i.

If there is a constant C such that $\| \eta_i \| \leq C$ for all i, then there exists some i such that $g(y_i) \leq M\delta + g^*(\frac{2}{2-\beta} - \frac{\beta}{2-\beta}\alpha)$ for any $\delta > 0$ and for some constant M.

Proof

First, we assert that there is some i such that

$$g(y_i) \leq \frac{\lambda_i^2 \bar{g}}{(\lambda_i^2 - 2\lambda_i)} - \frac{2\lambda_i g^*}{(\lambda_i^2 - 2\lambda_i)} - \frac{\delta}{(\lambda_i^2 - 2\lambda_i)}, \text{ for any } \delta > 0.$$

Set UBND ← cx.

If (UBND - LBND) $\leq \varepsilon$(UBND), then terminate with x optimal;

otherwise,

set $v_j \leftarrow -\hat{\pi}_{FROM(j)} + \hat{\pi}_{TO(j)} + c_j$, $j = 1,\ldots,2K$,

set $\eta \leftarrow (v_1 + v_{K+1},\ldots,v_K+v_{2K})$.

If Q = RFREQ, then set Q ← 0, set $\lambda \leftarrow \lambda_0$, set $\bar{\lambda}_k \leftarrow \lambda_0$

for $k = 1,\ldots,K$, and go to 3;

otherwise,

compute $\hat{\lambda}_k$ such that $0 \leq y_k - \hat{\lambda}_k\eta_k \leq \tilde{u}_k$, $k = 1,\ldots,K$,

set $\bar{\lambda}_k \leftarrow \min\{\bar{\lambda}_k/2,\hat{\lambda}_k\}$, $k = 1,\ldots,K$,

set $\lambda \leftarrow \min\{\bar{\lambda}_k, k=1,2,\ldots,K\}$.

Step 3 (Move to New Point)

3a. Set $y \leftarrow P[y - \lambda \frac{(UBND - LBND)}{\| \eta \|^2} \eta]$, set Q ← Q+1.

3b. Go to 2.

Note that the step size (iii) presented before may be rewritten for our function g as follows:

$$s_i = \frac{\lambda_i(g(y_i) - g^*)}{\| \eta_i \|^2}, \quad 0 < \lambda_i < 2,$$

where $\eta_i \in \partial g(y_i)$ and g^* is the optimal value of g.

where L,M are integers and $0 \leq L,M \leq K$. Then

$P(\hat{y}) = y^* = (0,\ldots,0,\hat{y}_{L+1},\ldots,\hat{y}_{L+M},\tilde{u}_{L+M+1},\ldots,\tilde{u}_K)$ is

a projection of $\hat{y}$ on S.

Following a description of the terminology used, our algorithm for obtaining an upper bound for P1 is presented below. Let RFREQ denote the frequency at which the constant λ in step size (iii) is reset to its initial value, λ_0, LBND denote a lower bound on P1, UBND denote an upper bound on P1, P denote the projection routine described in Proposition 10, ε denote the termination tolerance and Q denote the iteration count for the upper bound.

ALG 3 *UPPER BOUND ALGORITHM*

Step 1 (Initialization)

Choose $y \in S$.

Initialize LBND, RFREQ, ε, and λ_0.

Set $\tilde{u} \leftarrow (\min(u_1,u_{K+1}),\ldots,\min(u_K,u_{2K}))$.

Set $Q \leftarrow 0$, set $\bar{\lambda}_k \leftarrow \lambda_0$ for $k=1,\ldots,K$.

Step 2 (Find Subgradient and Step Size)

For allocation y, let $\bar{x}$ and $\hat{\pi}$, respectively, be the vectors of optimal primal and dual variables for

$\text{Min}\{\hat{c}\hat{x}: \hat{A}\hat{x} = \hat{b},\ 0 \leq \hat{x} \leq \hat{u}\}$. Construct x from $\bar{x}$,y.

for P4. Suppose the arc corresponding to j has "From" node j_1 and "To" node j_2. That is, arc j is the ordered pair (j_1,j_2). Then we define FROM(j) = j_1 and TO(j) = j_2. Using this notation, the following proposition gives the required formulae:

Proposition 8 (Shetty [26])

Let $\hat{\pi}$ be the vector of optimal dual variables for P4. Then $[\hat{\pi}, v_1, v_{K+1}, \ldots, v_K, v_{2K}]$ with $v_j = -\hat{\pi}_{FROM(j)} + \hat{\pi}_{TO(j)} + c_j$, $j=1,\ldots,2K$, are optimal duals for P3.

We now present two propositions that justify the projection routine used for the upper bound. The proofs may be found in Kennington and Helgason [21] and Shetty [26].

Proposition 9

Let S be a nonempty, convex set and $\hat{y} \notin S$. Then $y^* \in S$ is a projection of $\hat{y}$ on to S if $(\hat{y} - y^*)(y - y^*) \leq 0$ for all $y \in S$.

Proposition 10

Let $\hat{y} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_K) \in R^K$ with

$$\hat{y}_k < 0 \qquad \text{for } k = 1,\ldots,L$$

$$0 \leq \hat{y}_k \leq \tilde{u}_k, \ \tilde{u}_k = \min(u_k, u_{K+k}) \qquad \text{for } k = L+1,\ldots,L+M$$

$$\hat{y}_k > \tilde{u}_k \qquad \text{for } k = L+M+1,\ldots,K$$

$$
\begin{array}{lll}
\text{Minimize} & cx & \\
\text{s.t.} & Ax = b & (\pi) \\
& x_1 = y_1 & (v_1) \\
& x_{K+1} = y_1 & (v_{K+1}) \\
& \vdots & \\
& x_K = y_K & (v_K) \\
& x_{2K} = y_{2K} & (v_{2K}) \\
& 0 \leq x \leq u & (\mu).
\end{array}
$$

Then $\eta = (v_1 + v_{K+1},\ldots,v_K + v_{2K})$ is a subgradient of g at $y = (y_1,\ldots,y_K)$. As a result of Proposition 7, a subgradient at any given point $y = (y_1,\ldots,y_K) \in S$ required in our specialization of ALG 1 can be obtained by solving $\text{Min}\{cx: Ax = b, x_1 = y_1,\ldots,x_{2K} = y_K, 0 \leq x \leq u\}$, which we shall refer to as P3. After substituting $x_1 = y_1,\ldots,x_{2K} = y_K$, in $Ax = b$, we obtain a pure network problem, which we shall refer to as P4 and is given below:

$$
\begin{array}{ll}
\text{Minimize} & \hat{c}\hat{x} \\
\text{s.t.} & \hat{A}\hat{x} = \hat{b} \\
& 0 \leq \hat{x} \leq \hat{u}.
\end{array}
$$

To apply ALG 1, we need a procedure for constructing the optimal dual variables $(v_1,v_{K+1},\ldots,v_K,v_{2K})$ for P3 from the optimal dual variables

IV. THE UPPER BOUND

An alternate formulation of P1, which will be referred to as P2, is as follows:

Minimize $g(y)$

s.t. $y \in S$

where for any vector $y = [y_1,\ldots,y_K]$,

$$g(y) = \text{Min}\{cx: Ax = b,\ 0 \leq x \leq u,\ x_k = x_{K+k} = y_k \text{ for all } k\}$$

and

$$S = \{y: 0 \leq y_k \leq \min(u_k, u_{K+k}) \text{ for all } k\}.$$

Clearly, P1 and P2 are equivalent. That is, given an optimum for one, we can construct an optimum for the other. We will now show that P2 is a special case of the nonlinear program P0 and may be solved using the Subgradient Optimization Algorithm, ALG 1.

Proposition 6 (Shetty [26])

The real valued function g is piece-wise linear convex over the compact, convex and nonempty set S.

To apply the subgradient algorithm, we need a procedure for obtaining a subgradient of g at a point y. The following proposition shows that the dual variables may be used to construct a subgradient.

Proposition 7 (Shetty [26])

Let $(\pi, v_1, v_{K+1},\ldots,v_K, v_{2K}, \mu)$ be the optimal dual variables for

Step 2 (Find Subgradient)

Let $\bar{x} = [\bar{x}_1, \ldots, \bar{x}_n]$ solve

$$h(w) = \text{Min}\{cx + \sum_{k=1}^{K} w_k(x_k - x_{K+k}) : Ax = b,\ 0 \leq x \leq u\}.$$

Set LBND $\leftarrow h(w)$.

If (UBND - LBND) $\leq \varepsilon$(UBND) then terminate;

otherwise, set $d \leftarrow [(\bar{x}_1 - \bar{x}_{K+1}), \ldots, (\bar{x}_K - \bar{x}_{2K})]$.

Step 3 (Move to a New Point)

3a. Set $w \leftarrow w + pd$, set $p \leftarrow p/2$.

3b. Go to 2.

subgradient optimization technique for concave functions. This technique is similar to ALG 1 with a modification. Let $p_0, p_1, p_2, \ldots$ denote a sequence of step sizes and let $d_i \in \partial h(w_i)$. Then step 2 is replaced by:

Step 2 (Move to New Point)

Set $w_{i+1} \leftarrow w_i + p_i d_i$, set $i \leftarrow i+1$ and return to step 1.

To use this algorithm h(w) must be concave, and we need a means of generating subgradients. These two results follow:

Proposition 4 (Shetty [26])

The real valued function h is concave over R^K.

Proposition 5 (Shetty [26])

For a given $\bar{w}$, let $\bar{x}$ be an optimal solution to

$$\text{Min}\{cx + \sum_{k=1}^{K} \bar{w}_k(x_k - x_{K+k}) : Ax = b,\ 0 \leq x \leq u\}.$$

Then $d = [(\bar{x}_1 - \bar{x}_{K+1}), \ldots, (\bar{x}_K - \bar{x}_{2K})]$ is a subgradient of h at $\bar{w}$.

We used scheme (i) for step sizes. Let UBND denote an upper bound and assume that the optimal objective value is positive. Our algorithm for obtaining lower bounds is presented below:

ALG 2 LOWER BOUND ALGORITHM

Step 1 (Initialization)

Initialize UBND, step size p, and tolerance ε.

Set $w \leftarrow 0$.

be used to solve a network problem as well as reoptimize after problem data changes. Based on NETFLO [21], this code allows the user to change costs, bounds and/or requirements for a network problem and reoptimize. The tuning parameters used in all runs were as follows: ITERL = 5, ITERU = 10, REFREQ = 5, $\lambda_0 = 0.75$, $p_0 = 0.01$, and $\varepsilon = 0.1$. MODFLO [1] is used to reoptimize after each change to either the costs or right-hand-sides.

6.2 The Test Problems

The program NETGEN, a generator for large-scale network test problems, was used to generate 35 test problems. The parameters used to generate these problems are described in Klingman, Napier, and Stutz [22]. The test problems have between 200 and 1500 nodes, and 1500 and 7000 arcs. For each problem, the first 150 arcs were paired to form equal flow sides constraints. The characteristics of these test problems are listed in Table 1.

Our algorithm requires upper bounds on all equal flow arcs. Though NETGEN generates bounds on some of these arcs, there were others with no upper bounds. We set the maximum of all supplies and demands to be the upper bounds on such arcs. These bounds were acceptable since the optimal solutions obtained for our pure network problems were the same as the ones listed in NETGEN. Furthermore, for all 35 test problems the first 150 arcs were used to form 75 pairs of equal flow side constraints. We were unable to experiment with more than 75 pairs due to a core storage limitation of 301K. NETSIDE required approximately 300K octal words of storage for

problems 28 through 35 with 75 pairs and any further increase in the number of pairs would exceed the storage limitation.

6.3 Computational Results

All 35 test problems were solved on the CDC 6600 at Southern Methodist University, using the FTN compiler with OPT = 2. All 35 problems were solved twice using EQFLO; once with the same step size for every pair of equal flow arcs and the second time with different step sizes for different pairs. While using EQFLO to solve these problems, ALG 4 was followed exactly the first time, whereas, the computation of the step size for the upper bound was altered the second time. The modification was as follows:

Step 3 (Move to New Point)

$$3a.\quad \text{Set } y_k \leftarrow P\left[y_k - \bar{\lambda}_k \frac{(UBND-LBND)}{\|\eta\|^2}\eta_k\right],\ k = 1,\ldots,K,$$

set $Q \leftarrow Q+1$.

Note that this modification results in different step sizes for different equal flow pairs. The details of all runs are given in Tables 2 and 3. The times are in CPU seconds and exclude input and output.

The value 0.01 used for p_0 worked well for all test problems except problem number 11. This problem experienced difficulties in converging within 10% of the optimal. However, the problem did converge within 10%

of the optimum when we changed p_0 to values between 3 and 10.

The computational results presented in Tables 2 and 3 are summarized in Table 4. Letting T(ALG) denote the CPU time required to solve the 35 test problems using code ALG, the relationship is given below:

T(NETSIDE) = 2.54 (EQFLO), same step size,

T(NETSIDE) = 3.00 (EQFLO), different step sizes.

Note that EQFLO performs better as the problem size increases. Although EQFLO was slightly slower than NETSIDE on problems 1 to 10, its performance increased substantially on problems 11 to 35. In particular, EQFLO ran approximately 5 to 6 times faster than NETSIDE on problems 28 through 35 and these problems are fairly large. We expect EQFLO to perform even better on much larger problems. This is attributable to the fact that the time for pricing and updating increases dramatically for NETSIDE with an increase in the size of the network, whereas, the time increase should be relatively small for EQFLO because the above operations are performed very efficiently using labelling procedures on the rooted spanning tree.

The 75 side constraints made the problems approximately three times harder. That is, the pure networks were solved in 693 seconds while it required 1973 seconds to solve the equal flow problem. Klingman, Napier, and Stutz [22] solved the same 35 pure network problems in approximately 200 seconds using an advance start on a CDC 6600 at the

University of Texas at Austin. Richard Barr's best time on these 35 test problems is 104 seconds using ARC II [3]. This difference in time is due to the fact that EQFLO is a real code (as opposed to all-integer), uses an all artificial start, and does not use the advanced data structure or candidate list incorporated in ARC II.

These 35 problems were the largest that could be solved using NETSIDE under a core storage limitation of 301K octal words. However, EQFLO required much less storage; approximately 50% less than NETSIDE. This additional storage for NETSIDE results from the working basis inverse and the arrays required during the reinversion process.

VII. SUMMARY AND CONCLUSIONS

This paper presents a new procedure for the equal flow problem. Unlike the simplex method for the network problem with side constraints, this new procedure does not require a working basis. We have showed that using the subgradient optimization technique, the equal flow problem may be solved as two sequences of pure network problems. One sequence corresponds to a lower bound while the other corresponds to an upper bound. In the lower bound, each network differs from the previous one in that the cost vector has changed. In the upper bound, each network differs from the previous one in that the right hand side has changed. While solving the pure network problems with these changes in the problem data, a reoptimization procedure is used to obtain a good starting solution. Our technique terminates when the difference between two bounds is within a prespecified tolerance.

Subgradients for upper bounds are computed using the optimal dual variables obtained by solving the pure network problems. The subgradients for lower bounds are the difference between the flows on the equal flow arcs, obtained while solving the Lagrangean relaxation. The projection operation is easily implemented. The step sizes (i) and (iii), described in Section II, are used for lower and upper bounds, respectively. For these step sizes, we are guaranteed a solution at which the objective value is arbitrarily close to the optimal objective value.

We solved all test problems twice; once with the same step size for

all equal flow pairs, and once with different step sizes for each pair. The tests were conducted on a set of 35 randomly generated problems and a comparison was made with NETSIDE, a code that is designed to solve network problems with side constraints. On the average, our code ran approximately 3 times faster. However, it's performance improved substantially as the problem size increased. The new algorithm requires only 50% of the core storage required by NETSIDE.

Table 1 NETGEN Test Problems

Problem Number	Number of Nodes	Number of Arcs
Transportation Problems		
1	100 X 100	1511
2	100 X 100	1700
3	100 X 100	2207
4	100 X 100	2405
5	100 X 100	3100
6	150 X 150	3450
7	150 X 150	4800
8	150 X 150	5470
9	150 X 150	6395
10	150 X 150	6611
Assignment Problems		
11	200 X 200	1900
12	200 X 200	2650
13	200 X 200	3400
14	200 X 200	4150
15	200 X 200	4900
Capacitated Network Problems		
16	400	1374
17	400	2511
18	400	1374
19	400	2511
20	400	1484
21	400	2904
22	400	1484
23	400	2904
24	400	1398
25	400	2692
26	400	1398
27	400	2692
Uncapacitated Network Problems		
28	1000	3000
29	1000	3500
30	1000	4500
31	1000	4900
32	1500	4492
33	1500	4535
34	1500	5257
35	1500	5880

Table 2 Comparison of NETSIDE and EQFLO on 35 Test Problems
(Same step size for every equal flow pair)

Problem Number	NETSIDE	EQFLO					
	Optimal Objective	Total Time	Time				% of Optimal at Termination	
			Total	Pure Network	Lower Bound	Upper Bound	Lower Bound	Upper Bound
1	2694547	30	50	7	19	24	98	108
2	2350637	24	58	7	23	28	95	105
3	1939836	27	127	9	55	63	99	110
4	1612265	33	79	10	33	36	98	108
5	1480741	33	40	12	13	15	97	108
6	2472907	71	46	22	9	15	98	108
7	2236784	96	59	28	14	17	97	107
8	2223900	84	58	32	11	15	99	108
9	1839835	115	44	36	6	2	98	105
10	2291942	105	79	36	19	24	96	106
11*	4992	135	51	17	11	23	98	108
12	3573	105	93	23	20	50	95	105
13	3142	103	78	27	16	35	98	108
14	2787	118	34	31	1	2	99	101
15	2795	150	127	35	31	61	97	108
16	82161432	43	8	6	1	1	99	107
17	45601025	66	13	8	4	1	99	105
18	81600312	40	8	6	1	1	99	106
19	45601025	66	12	8	3	1	99	102
20	74065202	40	9	6	2	1	99	108
21	40137087	44	11	8	2	1	99	101
22	73429862	32	8	6	1	1	99	109
23	39354594	33	11	8	2	1	99	101
24	85926653	91	7	3	3	1	98	104
25	58203746	66	9	5	3	1	99	101
26	74267081	65	6	3	2	1	97	102
27	47295659	57	7	4	2	1	99	107
28	131316225	201	31	20	9	2	99	107
29	113594497	260	167	25	72	70	98	107
30	90569484	337	243	23	111	109	91	106
31	84943754	296	44	24	16	4	99	109
32	180390305	529	80	48	25	7	98	109
33	205246112	453	83	47	23	13	98	108
34	166247998	477	95	51	24	20	96	106
35	163964307	503	68	52	11	5	99	107

* $p_0 = 10.$

Table 3 Comparison of NETSIDE and EQFLO on 35 Test Problems
(Different step sizes for different pairs)

Problem Number	NETSIDE		EQFLO					
	Optimal Objective	Total Time	Time				% of Optimal at Termination	
			Total	Pure Network	Lower Bound	Upper Bound	Lower Bound	Upper Bound
1	2694547	30	47	7	16	24	97	108
2	2350637	24	50	7	19	24	94	105
3	1939836	27	105	9	42	54	99	109
4	1612265	33	74	10	29	35	98	108
5	1480741	33	37	12	10	15	95	106
6	2472907	71	46	22	9	15	98	107
7	2236784	96	57	28	13	16	97	105
8	2223900	84	42	32	4	6	98	109
9	1839835	115	44	36	6	2	98	105
10	2291942	105	76	36	19	21	96	105
11*	4992	135	53	17	9	27	94	104
12	3573	105	79	23	14	42	95	105
13	3142	103	63	27	11	25	98	108
14	2787	118	34	31	1	2	99	101
15	2795	150	108	35	24	49	98	107
16	82161432	43	8	6	1	1	99	107
17	45601025	66	13	8	4	1	99	105
18	81600312	40	8	6	1	1	99	106
19	45601025	66	12	8	3	1	99	102
20	74065202	40	9	6	2	1	99	108
21	40137087	44	11	8	2	1	99	101
22	73429862	32	8	6	1	1	99	109
23	39354594	33	11	8	2	1	99	101
24	85926653	91	7	3	3	1	98	104
25	58203746	66	9	5	3	1	99	101
26	74267081	65	6	3	2	1	97	102
27	47295659	57	7	4	2	1	99	107
28	131316225	201	31	20	9	2	99	107
29	113594497	260	81	25	28	28	97	107
30	90569484	337	148	23	62	63	93	104
31	84943754	296	44	24	16	4	99	109
32	180390305	529	80	48	25	7	98	109
33	205246112	453	78	47	22	9	98	108
34	166247998	477	86	51	23	12	97	107
35	163964307	503	68	52	11	5	99	107

* $p_0 = 10$.

Table 4 Summary of Computational Results

Problems	Time(NETSIDE)	Time(EQFLO)							
		Same step size				Different step sizes			
		Total	Pure Network	Lower Bound	Upper Bound	Total	Pure Network	Lower Bound	Upper Bound
1 - 10	618	640	199	202	239	578	199	167	212
11 - 15	611	383	133	79	171	337	133	59	145
16 - 27	643	109	71	26	12	109	71	26	12
28 - 35	3056	811	290	291	230	616	290	196	130
Total	4928	1943	693	598	652	1640	693	448	499

REFERENCES

1. Ali, A., E. Allen, R. Barr, and J. Kennington, "Reoptimization Procedures for Bounded Variable Primal Simplex Network Algorithms", Technical Report 83-OR-2, Department of Operations Research and Engineering Management, Southern Methodist University, Dallas, Texas, 75275, (1983).

2. Barr, R., K. Farhangian, and J. Kennington, "Networks with Side Constraints: An LU Factorization Update", Technical Report 83-OR-4, Department of Operations Research and Engineering Management, Southern Methodist University, Dallas, Texas, 75275, (1983).

3. Barr, R., F. Glover, and D. Klingman, "Enhancements of Spanning Tree Labelling Procedures for Network Optimization", INFOR, 17, 16-34, (1979).

4. Bazaraa, S., and C. Shetty, Nonlinear Programming: Theory and Algorithms, John Wiley & Sons, New York, N.Y., (1978).

5. Beck, P., L. Lasdon, and M. Engquist, "A Reduced Gradient Algorithm for Nonlinear Network Problems", ACM Transactions on Mathematical Software, 9, 57-70, (1983).

6. Carraresi, P., and G. Gallo, "Network Models for Vehicle and Crew Scheduling", European Journal of Operations Research, 16, 139-151, (1984).

7. Charnes, A., and W. Cooper, Management Models and Industrial Applications of Linear Programming, Volume II, John Wiley & Sons, New York, N.Y., (1961).

8. Chen, S., and R. Saigal, "A Primal Algorithm for Solving a Capacitated Network Flow Problem with Additional Linear Constraints", Networks, 7, 59-79, (1977).

9. Fletcher, R., and C. Reeves, "Function Minimization by Conjugate Gradients", Computer Journal, 7, 149-154, (1964).

10. Glover, F., and D. Klingman, "The Simplex SON Algorithm for LP/Embedded Network Problems", Mathematical Programming, 15, 148-176, (1981).

11. Glover, F., R. Glover, and F. Martinson, "The U. S. Bureau of Land Managements's New NETFORM Vegetation Allocation System", Technical Report of the Division of Information Science Research, University of Colorado, Boulder, Colorado, 80309, (1982).

12. Goffin, J., "On Convergence Rates of Subgradient Optimization Methods", Mathematical Programming, 13, 329-347, (1977).

13. Griffith, R., and R. Stewart, "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems", Management Science, 7, 379-392, (1964).

14. Grigoriadis, M., and W. White, "A Partitioning Algorithm for the Multicommodity Network Flow Problem", Mathematical Programming, 3, 157-177, (1972).

15. Hartman, J., and L. Lasdon, "A Generalized Upper Bounding Algorithm for Multicommodity Network Flow Problems", Networks, 1, 333-354, (1972).

16. Helgason, R., and J. Kennington, "Spike Swapping in Basis Reinversion", Naval Research Logistics Quarterly, 4, 697-702, (1980).

17. Helgason, R., "A Lagrangean Relaxation Approach to the Generalized Fixed Charge Multicommodity Minimal Cost Network Flow Problem", Unpublished Dissertation, Department of Operations Research and Engineering Management, Southern Methodist University, Dallas, Texas, 75275, (1980).

18. Hellerman, E., and D. Rarick, "Reinversion with the Preassigned Pivot Procedure", Mathematical Programming, 1, 195-216, (1971).

19. Kaul, R., "An Extension of Generalized Upper Bounded Techniques for Linear Programming", ORC Report No. 65-27, Department of Operations Research, University of California, Berkeley, California, (1965).

20. Kennington, J., "Solving Multicommodity Transportation Problems Using a Primal Partitioning Simplex Technique", Naval Research Logistics Quarterly, 24, 309-325, (1977).

21. Kennington, J., and R. Helgason, Algorithms for Network Programming, John Wiley & Sons, New York, N.Y., (1980).

22. Klingman, D., A. Napier, and J. Stutz, "NETGEN: A Program for Generating Large Scale Minimum Cost Flow Network Problems", Management Science, 20, 814-821, (1974).

23. Poljak, B., "Minimization of Unsmooth Functionals", U.S.S.R. Computational Mathematics and Mathematical Physics, 9, 14-29, (1969).

24. Sakarovitch, M., and R. Saigal, "An Extension of Generalized Upper Bounding Techniques for Structured Linear Programs", SIAM Journal of Applied Mathematics, 15, 906-914, (1967).

25. Shepardson, F., and R. Marsten, "A Lagrangean Relaxation Algorithm for the Two Duty Period Scheduling Problem", Management Science, 26, 274-281, (1980).

26. Shetty, B., "The Equal Flow Problem", unpublished dissertation, Department of Operations Research, Southern Methodist University, Dallas, Texas, 75275, (1985).

27. Shorn, N., "On the Structure of Algorithms for the Numerical Solution of Optimal Planning and Design Problems", Dissertation, Cybernetics Institute, Academy of Sciences, U.S.S.R., (1964).

28. Turnquist, M., and C. Malandraki, "Estimating Driver Costs for Transit Operations Planning", Presented at the Joint National Meeting of ORSA/TIMS, Dallas, (1984).

29. Zangwill, W., Nonlinear Programming: A Unified Approach, Prentice Hall, Englewood Cliffs, New Jersey, (1969).

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CCS 511	2. GOVT ACCESSION NO. AD-A159 594	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Equal Flow Problem		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) I. Ali, J. Kennington, B. Shetty		8. CONTRACT OR GRANT NUMBER(s) N00014-82-K-0295
9. PERFORMING ORGANIZATION NAME AND ADDRESS Center for Cybernetic Studies The University of Texas at Austin Austin, TX 78712		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research (Code 434) Washington, D.C.		12. REPORT DATE April 1985 13. NUMBER OF PAGES 39
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Linear Programming, Network Models, Networks with Side Constraints, Equal Flow Problem

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper presents a new algorithm to solve a network problem with equal flow side constraints. The proposed solution technique is motivated by the desire to exploit the special structure of the side constraints and to maintain as much of the characteristics of pure network problems as possible. Not only has specialized software for the efficient solution of pure networks been developed, but the same computational efficacies lend themselves to the solution of sequences of minimum cost network flow problems by using reoptimization procedures. Our solution technique for the equal flow problem consists of solving

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract

two sequences of pure network problems. One sequence yields tighter lower bounds on the optimal value by considering the Lagrangean relazation of the equal flow problem in which the side constraints are dualized. The second sequence yields upper bounds on the optimal value for the problem and maintains a feasible solution at all times. This sequence is obtained by considering a reformulation of the equal flow problem based on parametric changes in the requirements vector. The procedure has the added attractive feature that it provides a feasible solution which is known to be within a percentage of the optimal at all times. As such, the algorithm terminates when a solution with a prespecified tolerance on the objective function value is obtained. On NETGEN problems, using the first 150 arcs to form 75 equal flow side constraints, we found that the new algorithm is approximately 3 times faster than existing techniques and requires only 50% of the storage.

END

FILMED

10-85

DTIC